Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 25 (2013) 348 – 359

Procedia Computer Science

2013 International Conference on Virtual and Augmented Reality in Education

Splicing of Concurrent Upper-Body Motion Spaces with Locomotion

Christos Mousas[a,b], Paul Newbury[a], Christos-Nikolaos Anagnostopoulos[b,*]

[a]*Department of Informatics, University of Sussex, Falmer House, Brighton BN1 9QH*
[b]*Department of Cultural Technology and Communication, University Hill, Mytilene 81100, Greece*

Abstract

In this paper, we present a motion splicing technique for generating concurrent upper-body actions occurring simultaneously with the evolution of a lower-body locomotion sequence. Specifically, we show that a layered interpolation motion model generates upper-body poses while assigning different actions to each upper-body part. Hence, in the proposed motion splicing approach, it is possible to increase the number of generated motions as well as the number of desired actions that can be performed by virtual characters. Additionally, we propose an iterative motion blending solution, inverse pseudo-blending, to maintain a smooth and natural interaction between the virtual character and the virtual environment; inverse pseudo-blending is a constraint-based motion editing technique that blends the motions enclosed in a tetrahedron by minimising the distances between the end-effector positions of the actual and blended motions. Additionally, to evaluate the proposed solution, we implemented an example-based application for interactive motion splicing based on specified constraints. Finally, the generated results show that the proposed solution can be beneficially applied to interactive applications where concurrent actions of the upper-body are desired.

Selection and peer-review under responsibility of the programme committee of the 2013 International Conference on Virtual and Augmented Reality in Education

Keywords: character animation; concurrent actions; inverse pseudo-blending; motion editing; motion synthesis, motion splicing

1. Introduction

In their everyday lives, humans perform actions that involve concurrent upper-body tasks performed both while stationary and moving. Similarly, virtual characters should be able to perform simultaneous actions, as

* Corresponding author.
E-mail address: c.mousas@sussex.ac.uk ; p.newbury@sussex.ac.uk ; canag@ct.aegean.gr

Selection and peer-review under responsibility of the programme committee of the 2013 International Conference on Virtual and Augmented Reality in Education
doi:10.1016/j.procs.2013.11.042

for example when a character is switching on the light while opening a door. Although examples like this are fairly simple, the ability to generate motions for concurrent tasks that are performed simultaneously, or that are separated by a small delay, is quite important, as it gives the virtual character the ability to perform desired actions naturally. Hence, unlike current motion synthesis techniques, where the ability to generate motions is based on a walk-stop-action-walk actions, the proposed solution circumvents the stopping approach by generating the desired actions on-the-fly, simultaneously with other actions.

As it is desired for the character to be able to engage in various concurrent upper-body actions and simultaneous locomotion sequences assigned to lower-body parts, and also for the character to be able to generate different actions with both the right and left hands, it is necessary to handle the motion capture data in such a way as to generate the desired results. In past years, a variety of approaches for handling and synthesising new motion sequences based on specified body parts have been proposed. In the technique known as motion splicing [1], the motions of the various body parts are spliced so as to generate new motions.

In general, in the most common approach for handling and synthesising new motion sequences using motion splicing, the motion space for upper-body parts M_U is spliced with that for lower-body parts M_L. Using motion splicing, the total number of captured motions is reduced to M_U+M_L motion sequences, rather than capturing M_U*M_L motion sequences. Although, in the proposed solution, the concurrent actions generated by different upper-body tasks are spliced, such that M_U represents a splicing of the right and left sides of the upper-body such as $M_{U(right)}$ and $M_{U(left)}$, respectively. Hence, the number of required actions can be produced by $M_{U(right)}+M_{U(left)}+M_L$ rather than $M_{U(right)}*M_{U(left)}*M_L$. Thus, with the proposed solution, it is possible to generate more actions or more combinations of actions of the upper-body while locomotion- or non-locomotion-based sequences are assigned to the lower-body.

Thus, this paper presents a technique that splices the action spaces of right and left upper-body parts (as in the examples in Figure 1) by solving the motion synthesis problem of the upper-body main trunk using a layered interpolation motion model. Additionally, after presenting the basic methodology used in the proposed motion splicing approach (see Section 3), we examine the ability of the model to automatically generate the desired motions of each upper-body part based on specified constraints (see Section 4). Finally, we evaluate the proposed pseudo-blending approach by implementation testing (see Section 5).

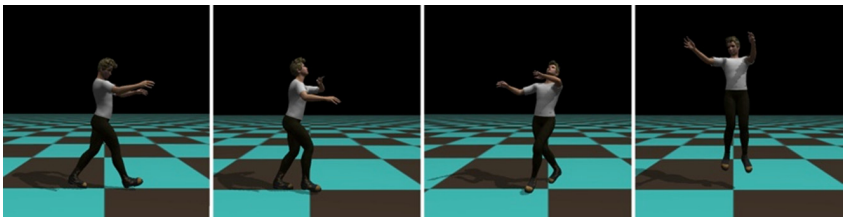

Fig. 1. Generated concurrent upper-body tasks based on specified constraints. Those motions generated simultaneously while the lowe-body performs locomotion tasks.

2. Related Work

During the past years, a variety of approaches have been proposed for simulating body movements using motion slicing. The basic advantage of combining the different motions of individual body parts is that the amount of captured or designed by the hand motion data required to generate a motion can be reduced. Thus, motion splicing techniques allow the splicing of concurrent upper-body motion spaces while the lower-body is performing locomotion movements [2] [3]. This approach was proposed by Ikemoto and Forsyth [4] for the splicing of limb movements, in which a motion capture database was developed that associates the motions of different limbs and which suggests rules for synthesising natural-looking motions. Solutions similar to the work

of Oshita [5] are able to generate motions of different body parts, which can then be combined by extracting the vibration of the integrated motions to generate a natural-looking sequence in which the temporal motion properties appear as a newly synthesized motion.

Although a number of different approaches to motion splicing have been proposed for capturing the data required to generate new motions, the solution proposed by Al-Ghreimil and Hahn [6] is particularly promising, as it subtracts the upper-body auxiliary motion M_U from M_L and tries to find similarities between the motions so as to determine differences between the joint trajectories. Ha and Han [7] designed new motions by developing an approximation technique that determines the sample positions of decoupled data in parameter space. Hence, in this case, motion sequence for different body-part can be combined, geneveting a vast amount of new motions. Similarly, the solution proposed by Jang et al. [8] splice the virtual character's body together from discrete parts, attempting to generate new motion sequences by using multiple motion-capture databases. Other motion splicing techniques, such as that proposed by Majkowska et al. [9], transplant hand gestures onto full body motions. Several other techniques have been developed that take into account body part dependencies, such as that of Ma et al. [10], who applied motion splicing to model motion variation, and Mousas and Newbury [18] who used motion layers for splicing the lowe-, upper-, and hand-body parts. Finally, Tamada et al. [11] and Ng et al. [12] designed splicing motions associated with a resultant motion derived from motion graphs, giving the ability to generate long animation sequences in which partial motions appear at specified time steps.

Although, most solutions, even if they do provide the ability to generate a vast number of new natural-looking motions by extracting necessary temporal in formation, from the best of our knowledge only a limited number of techniques, such as [7] [8], are able to splice actions that co-occur at similar hierarchical levels of the body, while simultaneously maintaining the naturalness of each motion as it is applied in the context of the whole body.

3. Overview

In this section, we present the separate components that must be computed to generate a motion using the concurrent motion splicing approach. Specifically, we present a technique to splice the virtual character's body in accordance with a spatial alignment process, which orients the character in the correct general direction, as well as a velocity-based time alignment process for generating the motions of each upper-body part, which maintains the temporal properties of the locomotion sequence. Additionally, to generate a natural-looking pose of the virtual character while two different actions are evolving concurrently, a layered interpolation model is presented that measures the influence of each integrated motion on the main trunk of the upper-body. Finally, the proposed motion blending technique ensures that the character is able to interact with the specified goal.

3.1. Motion Splicing

In this section, we present the motion splicing approach for handling a virtual character's motion. Specifically, the proposed motion splicing technique must be able to separate the virtual character's body into three parts, the left upper-body part, the right upper-body part, and the lower-body part, as illustrated in Figure 2. In the proposed solution, the first step divides the motion data into lower M_L and upper M_U motion spaces, and then a second step divides the upper motion space into right $M_{U(right)}$ and $M_{U(left)}$ motion spaces.

It should be noted that in the proposed method, the upper-body is divided into right and left parts, but the main upper-body trunk is included in both the right and left layers. This condition is imposed because, to deal with the different actions that are assigned to each upper-body part, and to integrate the final motion of the character, the generated motion should retain information related to both upper-body motions, and the final generated motion should represent the temporal motion properties of both upper-body parts. Hence, to generate a motion

sequence for the upper-body trunk, a layered interpolation method is presented (see Section 3.4) that tries to maintain all of the motion properties by computing at each time step how each motion influences the main upper-body trunk.

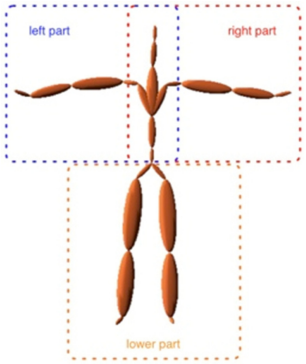

Fig. 2. The splicing strategy involves three body parts: the left upper-body, the right upper-body, and the lower-body part.

3.2. Spatial Alignment

While different motions can be assigned to both the right and left parts of the upper-body, it is necessary to align those motions spatially with respect to the general direction of the body. This spatial alignment is quite important to avoid incorrectly generated poses. Hence, in the pre-processing stage of the proposed solution, a spatial alignment process based on the solution of Heck et al. [2] is implemented, which aligns and integrates the motions of the left and right upper-body parts.

For the spatial alignment process, it is necessary that a motion is considered as a reference, and then the desired motion is rotated in such a way as to coincide and be integrated with the reference motion. Thus, a point cloud is formed between the reference motion of M_U and each integrated motion, which is then translated and rotated to coincide with the coordinate system of the pelvis. Finally, using the method proposed by Horn [13], the three-dimensional orientation q_r that minimises the sum of squared distances between the corresponding points can be found. Hence, the local pelvis orientation can be computed as $q_p * q_r$, where q_p is the local pelvis orientation of the reference sequence. Finally, it should be mentioned that the spatial alignment process is generated off-line so as to increase the computational capacity of the system.

3.3. Time Alignment

While a motion can be integrated with respect to either the left or right upper-body parts, it is necessary that the motion be warped in time so as to retain the temporal time variations of the rest of the body. However, time warping methods, such as those proposed by Heck et al. [2] and van Basten and Egges [3], cannot be implemented in our procedure because, in the proposed approach, the upper-body tasks are retrieved from non-locomotion sequences. Hence, we implemented a velocity-based time alignment method that measures the temporal velocity of the hand from the locomotion sequence M_L, and maps this velocity to $M_{u(i)}$, which is an upper-body action.

We first measure the displacement of the hand d_{hand} from the initial locomotion sequence between the time period $t_{hand}(i-1)$ and $t_{hand}(i)$. The hand displacement is measured at each Δt_{hand} by deriving the foot-state contact with the ground, using a height and velocity-based footstep detector which specifies the period during which the reaching should be generated. Then, it is possible to compute the hand velocity as $v_{hand}=\Delta d_{hand}/\Delta t_{hand}$. In the second step of the time alignment method, the velocity information of the corresponding hand retrieved from the

locomotion sequence is passed to the non-locomotion action; the displacement of the end-effector position of the hand is retrieved for the whole duration of the action, and the time is computed as $t_{U(hand)} = \Delta t_{hand} * \Delta d_{U(hand)} / \Delta d_{hand}$. Hence, having defined the time variation of one of the upper-body actions $M_{u(i)}$, the remaining actions are mapped linearly, so as to retain the same temporal timing variations for all the actions. Thus, while various locomotion sequences are evolving (e.g., simple walking to running), it is possible to generate time variations for each upper-body action that mimics the timing variation of the hand retrieved from the locomotion sequences.

3.4. Layered Interpolation Motion Model

While two different motions can be assigned to the right and left layers of the upper-body at any given time, it is necessary to examine the possibility of generating a natural-looking pose while the motions are evolving. Thus, the spliced motions, which are responsible for the upper-body actions, are interpolated separately, and the interpolated motions are assembled then together. Because a different motion capture database is required for each layer, and each layer can have a different influence on the remaining parts, it is necessary to examine how each motion resulting from the specified constraints influences the main upper-body trunk.

In general, motion synthesis of the upper-body main trunk can be separated into three basic steps. First, the most relevant motions responsible for the left body part layers should be extracted and then blended using the proposed motion blending technique (see Section 3.5) to retrieve the most suitable action $P_{L(G)}$. Second, the right body part layer motion should be extracted and parameterised by the extracted left hand motion (see Section 4.1), more specifically by the position and orientation of the root. Then, the most suitable motions are blended to retrieve the most suitable action $P_{R(G)}$. Third, the extracted motion of the upper-body is interpolated to retrieve the desired weights for each degree of freedom. Thus, for the calculation of the main upper-body trunk pose, as it should be the result of the goals defined by both hands, the final pose is estimated as a weighted interpolation function for each degree of freedom for both parts, according to:

$$M_{upperBodyn(n)} = w_1 * P_{left(n)} + w_2 * P_{right(n)} \tag{1}$$

where w_1 and w_2 are the weights computed for each degree of freedom of the upper-body motion. The task is solved as an optimisation problem following the solution of Wang and Xia [14], which can be represented as:

$$min\ \Sigma_{i=1,\ldots,s} = w_1 * P_{i,left(n)} + w_2 * P_{i,right(n)} - P_{i,body(n)} \tag{2}$$

where S denotes the total number of motion samples, $w_1 + w_2 = 1$, $P_{i,left(n)}$ and $P_{i,right(n)}$ are respectively the the i-th example pose of the main upper-body trunk synthesised from the i-th parameter of the left and right upper-body part, and $P_{i,body(n)}$ is the i-th sample pose of the main upper-body trunk. Finally, n denotes the n-th element of the corresponding vector $n=1,\ldots,d_{body}$, where d_{body} is the total degrees of freedom. The solution of this optimization problem can be retrieved by a least squares method, where the weighted variables at each time step, w_1 and w_2, are estimated according to:

$$w1w2 = P1,leftn \ldots P1,rightn \ldots PS,leftn1PS,rightn1 + \cdot P1,bodyn \ldots PS,bodyn1 \tag{3}$$

Thus, it is possible to retrieve the upper-body main trunk pose at each time step as the motions are evolving, resulting in a natural-looking pose of the character, as presented in Figure 3.

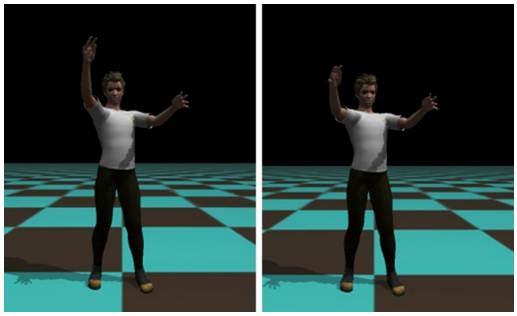

Fig. 3. Example poses generated by transplanting limbs (left), and the generated motion based on the layered interpolation motion model (right).

3.5. Inverse Pseudo-Blending

As our goal is to handle motion capture data to generate motions based on specified constraints, it is necessary to examine methods to blend the most suitable of those located in our motion capture database, thus giving the ability to generate motions to fulfil the desired constraints. Even though there are many motion blending techniques that can provide a desirable result, it is proposed an iterative method the inverse pseudo-blending (IPB). Unlike in the inverse blending approach [15], where given an end-effector position, using pseudosampling techniques generates the blended motion by minimising the error between the desired position and the blended motion's end-effector position, in the IPB approach, given the desired end-effector position, a new pseudo-position of the goal hull's vertex (that is influenced by a greater number of blendings) is rearranged to retrieve the desired result.

The IPB is a motion blending technique that allows interpolation of joint orientations. Its main advantage is that is easy to implement. However, as with other pseudo-blending techniques, such as that proposed by Kovar and Gleicher [16], IPB cannot provide an exact desired end-effector position; however, the results tend to be close to the desired outcome.

For the IPB, given a specified position of the end effector P_G, we generate the most suitable motions that enclose end effector in the desired goal, represented by a tetrahedron with vertices v_0, v_1, v_2 and v_3. Then, the values for each position are blended as $P_G=\Sigma_{i=1\ldots3}w_i*M_i$ producing the position of the end effector. However, blending of these motions does not produce the desired end-effector position. To achieve the correct position, horizontal and vertical lines that pass through the desired end-effector position segment the goal hull G into four different goal hulls, such that each one contains one of the vertices v_i (figure 4). Then, the vertex v_i, of the new generated goal hull G'_i that encloses the wrong end effector position, is rearranged so as to achieve the desired result.

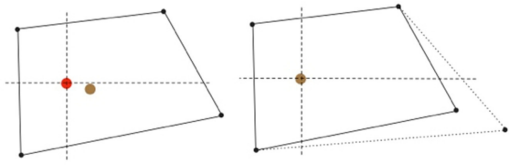

Fig. 4. The inverse pseudo-blending (IPB) approach. A generated goal hull showing the generated end-effector position (browd dot) and the desired end-effector position (red dot) (left), and the new goal hull after rearanging one of the vertices v_i to a new position, such that the desired and generated end-effector position coincide (right).

In the IPB approach, the new goal hull that is generated G'_i (i.e., that encloses the incorrect end-effector position P'_G) has too strong of an influence on the blending result. Thus, by changing the position of vertex v_i using a weighting variable w_i , G'_i will less strongly influence the generated position P'_G (see figure 4). To calculate the new position v'_i , the distance d_G between the end-effector position P_G and the generated end-effector position P'_G is minimised iteratively such as $(d_G=P_G-P'_G)$, and the new position is then calculated according to:

$$v'_i=v_i+d_G \tag{4}$$

However, in the iterative process, it is necessary to define the number of iterations that are required. By considering the initial distance d_{init} between P_G and P'_G and the distance retrieved after each iteration $d_{init(i)}$, the iteration is continued until a minimum displacement is achieved; we selected a minimum displacement value of 1%, such that the iteration terminates when $d_{init(i)}/d_{init}<.01$. On the other hand, as the iterations evolve, it is possible that P'_G is relocated to place in which it is enclosed by another tetrahedron G'_i . Nonetheless, the iterations continue until the displacement value reaches the desired minimum. Results showing the number of required iterations based on various generated actions are presented in Section 5.3. Finally, it should be noted that while the IPB was used here to compute reaching task, the same approach can be used for various other tasks where the motion blending approach for constraint-based motions is desired.

4. Concurrent Action Synthesis

In the previous section, we presented the most important components required to generate concurrent upper-body actions. In this section, we present the integration of lower-body computations into the method, as it is desired that the virtual character to be able to perform upper-body actions while a lower-body locomotion sequence is evolving.

4.1. Action Space

When a character is trying to reach two different goal positions with his hands while simultaneously moving, it is necessary to examine the actual three-dimensional space in which each action is performed. Using a motion capture database in which actions of the upper-body left and right parts are subsets of the upper-body task, $M_{L(i)} \subset M_{L(U)}$ and $M_{R(i)} \subset M_{R(U)}$, respectively, we generate two Delaunay tetrahedrons such that each tetrahedron is responsible for an upper-body part layer, and where the vertices $v_{L(i)}$ of the tetrahedron correspond to the motion $M_{L(i)}$. As the character performs a locomotion sequence, the actual tasks that are generated should be parameterised in such a way that the character interacts with the desired tasks. Hence, each motion $M_{L(i)}$ is parameterised based on the actual spine position of M_U as $f(M_{L(i)})$.

On the other hand, as actions are assigned to both upper-body layers, the actual tasks must be parameterised in such a way as to avoid incorrect end-effector positions. For example, if the character is asked to perform a left-hand goal position $P_{L(G)}$, then the Delaunay tetrahedron for the other hand should be parameterised on the basis of the pelvis orientation generated by $d_{L(G)}$, as illustrated in Figure 5. Hence, each tetrahedron must fulfil the form $f(M_U,q_p)$, where P_q is the displacement orientation of the pelvis defined by a task assigned either to the left or right hand.

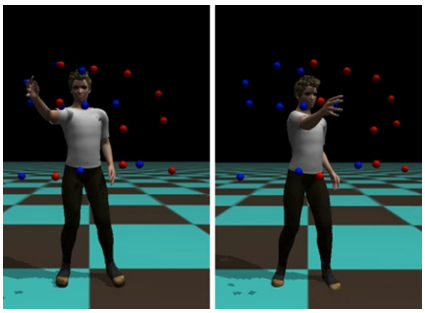

Fig. 5. The generated goal hulls for both hands (left), and an example of goal-hull parameterisation based on pelvis orientation while the right hand performs an action (right).

4.2. Splice on Action

Having defined the method for generating the actual action space in which the character can perform the desired actions, it is next necessary to examine the method for switching on the splicing technique that is used to generate the desired actions. Thus, as a locomotion procedure evolves, it is necessary to define the time period during which each upper-body motion sequence is generated, thus giving the virtual character the ability to successfully perform the specified action. However, the splicing of the character's actions depends on the three-dimensional position of the constraint and its accordance with the time required to perform the specified action. Thus, to avoid any errors in the timing of a sequence, it is necessary to define the metric with which an action is generated.

Considering a locomotion sequence in which the position at each time step of the root is $P(t)$, we first examine the time $t_{P(t)}$ at which the distance of the desired task from the root is at a minimum; this time is considered to be the time at which the character reaches the defined goal. In the second step, after having computed the timing variation $t_{U(hand)}$ of the desired action based on the locomotion sequence (see section 3.3), it is possible to retrieve the splice on the action corresponding to the time period, $t^{s}_{P(t)} = t_{P(t)} - t^{U}_{hand}$ (see Figure 6). Hence, in the proposed solution, where actions are assigned to both hands, the time period for generating each action is computed separately so as to accurately generate each motion.

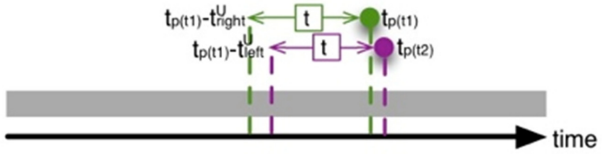

Fig. 6. The response time based on constraints.

4.3. Constraint-Action Generation

Having defined the actual tasks that the character is performing with both hands on the basis of the action space (Section 4.1), we next examine the motion parameterisation process that allows the character to reach the desired goal. Hence, given a goal position $P_{L(G)}$ for the left hand, the vertices based on the reference motions $M_{L(i)}$ that enclose $P_{L(G)}$ generate a tetrahedron that can be used for blending the motions, such as $\Sigma_{i=0,\dots,3} w_{L(i)} * f(M_{L(i)})$ where $\Sigma_{i=0,\dots,3} w_{L(i)} = 1$. Similarly, to retrieve the most suitable motion for the right hand by the time the left hand has been assigned a task, the motion editing process must take into account the parameterisation of the root-generated tetrahedron, as $\Sigma_{i=0,\dots,3} w_{L(i)} * f(M_{R(i)}, q_p)$. Finally, as mentioned in section

3.4, having defined the blending weighs, the pseudo-blending approach is used to retrieve the final positions $P_{L(G)}$ and $P_{R(G)}$.

5. Implementation and Results

We here present the results of experiments to test the proposed method of generating concurrent actions of the upper-body while the lower-body is performing locomotion motions. The results of motion editing based on various actions assigned to the upper-body parts, either as simple generated motions or as constraint-based motions, are presented. Additionally, the performance of the system, as well as the results related to the proposed motion blending technique, are presented. We used motion capture data retrieved from CMU motion capture database [17] for both upper- and lower-body actions. We should note that the motion capture data were downsampled to 60 fps and the experiments were performed on a system with an Intel i7 processor operating at 2.2 GHz and with 8 GB of RAM.

5.1. Concurrent Motion Generation

For the experimental testing, the task for the virtual character was either to reach two different targets (one with each hand) or to reach one target with one hand while the other hand was performing other actions. Thus, various actions related to reaching, punching, knocking on a door, and drinking water were assigned to both the right and left upper-body parts during locomotion sequences such as walking, running, and stepping. The system automatically tries to generate a layered interpolation motion model so as to provide a natural-looking pose of the upper-body main trunk.
Additionally, in order to evaluate the proposed approach, the character was constrained by the types of required actions, and the placement of the targets at different positions and heights. Depending on the three-dimensional position of the desired target assigned to each hand, it is possible to generate motions for both hands that are either synchronous or asynchronous, depending on the actual characteristics of the targets. Examples of generated actions are illustrated in figure 7.

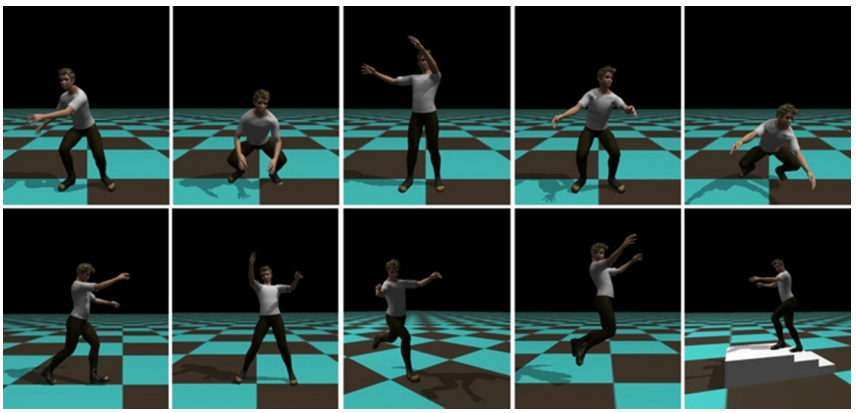

Fig. 7. Examples of constraints imposed by reaching motions on concurrent upper-body tasks generated while not moving (upper row), and while moving (lower row).

Furthermore, to generalise the proposed solution, the method compared the ability to retarget the motion to characters of different heights (see Figure 8) reaching for the same targets with both hands in the Cartesian space. Although, while the characters have different heights it is necessary to be examined the actual tasks that

both characters can perform. Hence, during retargeting it is computed the intersection of actions that can be generated based on the similarities that the generated goal hulls of both characters have.

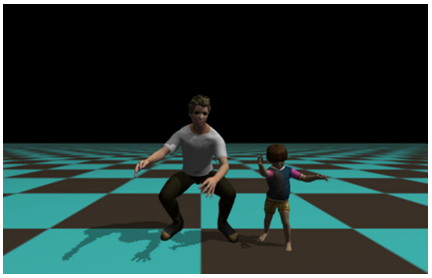

Fig. 8. Characters of different heights reaching for the same target

In the final evaluation step, the methodology was evaluated against the solution proposed by Ikemoto and Forsyth [4], which assigned two different motions to both limbs. Thus, as illustrated in Figure 9, the approach proposed in this study provided the ability to generate a better pose for the whole body and a better pose for the main trunk while two different motion sequences were assigned to the right and left sides.

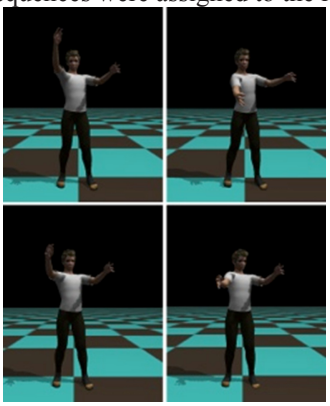

Fig. 9. Generated motions based on Ikemoto and Forsyth (left column), and the proposed approach (right column)

5.2. Performance Evaluation

In the implementation, the performance of the system is enhanced by a spatial alignment process that is generated in advance of each motion. Hence, the information provided by point clouds for the spatial alignments of the lower-body locomotion sequence and each upper-body part are retrieved in real time using dynamic programming. On average, the time required for the computational processes was 12% for the time alignment (Section 3.3), 52% for the layered interpolation motion model (Section 3.4), and 36% for the motion blending (Section 3.5).

5.3. Inverse Pseudo-Blending Evaluation

We here evaluate the proposed motion blending approach. Specifically, our motion blending technique is evaluated in terms of the number of iterations required to provide desirable results, by minimising the distance between the desired target P_G and the blending targe P'_G. We assigned 20 different target positions and measured

the iterations necessary to achieve a percentage displacement $d_{G(i)}/d_G<.01$ (i.e., >1%). Figure 10 shows the percentage displacement in the generated examples. On average, seven iterations are necessary to retrieve a desirable result. Finally, Figure 11 shows the generated end-effector positions after successive iterations.

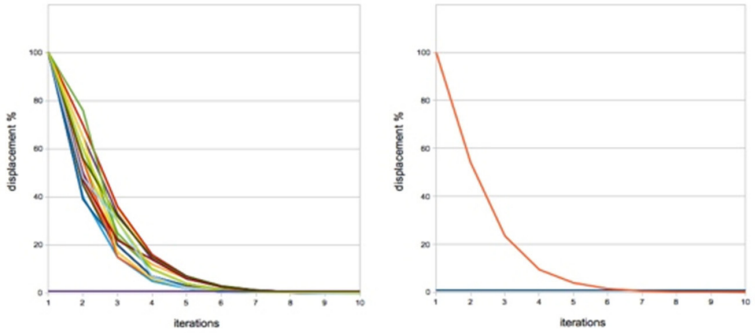

Fig. 10. The percentage displacement of the desired end-effector position as a function of the number of iterations for 20 randomly generated example (right), and the average number of iterations for all the generated results (left). The coloured horizontal line denotes the minimum displacement value.

Fig. 11. Resulting blends and goal hull rearrangements at successive iterations i={0,1,3,5,7}. The yellow sphere denotes the desired end-effector position.

6. Conclusion

This paper presents an approach for handling a virtual character's motion based on assigning simultaneous concurrent actions to upper-body parts and locomotion actions to the lower-body, using motion splicing to generate the motions. In the proposed solution, we examine not only the ability of the solution to generate new motion sequences based on a few specified constraints, but also the possibility of extending existing motion splicing approaches by adding methods for generating concurrent upper-body actions. In addition to the motion synthesis process, a layered interpolation motion model attempts to generate a natural-looking motion of the upper-body main trunk while assigning two different motions to the right and left upper-body parts. Additionally, the inverse pseudo-blending technique is proposed, which allows the generation of desired tasks in cases in which interactive motion parameterisation is desired. Moreover, the proposed implementation shows that the ability to generate motions based on a motion splicing approach can increase the number of possible actions that can be generated. Finally, because the method is based on interactive frame rates, it seems ideal for future implementation in applications where users require the ability to generate desired actions interactively.

On the other hand, in the proposed solution, since the motions that are used have been retrieved from motion capture data, it can be assumed that the synthesised motions are physically correct and incorporate all of the properties required of the proposed solution, such as the necessary motion dynamics relationships. Hence, we

did not examine the ability of the method to extract and pass any physical characteristics of the initial motion sequence to the generated motions. However, this assumption should be examined more closely, as each motion can have distinct properties that should be taken into account. Thus, as the need to design more realistic motions increases, it may be necessary to incorporate motions retrieved from different individual body parts, and to examine whether future implementation of integrated motion approaches can provide the information required for designing high-quality motion sequences.

References

[1] van Basten BJH, Egges A. Motion transplantation techniques: a survey. *IEEE Computer Graphics and Application* 2012; **32** (3):16–23.
[2] Heck R, Kovar L, Gleicher M. Splicing upper-body action with locomotion. *Computer Graphics Forum* 2006; **25**(3):459-466.
[3] van Basten BJH, Egges A. Flexible splicing of upper-body motion spaces on locomotion. *Computer Graphics Forum* 2011; **30** (7):1963-1971.
[4] Ikemoto L, Forsyth DA. Enriching a motion collection by transplanting limbs. Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2004, pp. 99-108.
[5] Oshita M. Smart motion synthesis. *Computer Graphics Forum* 2008; **27** (7): 1909-1918.
[6] Al-Ghreimil N, Hahn J. Combined partial motion clips. Proc. of WSCG, 2003.
[7] Ha D, Han HJ. Motion synthesis with decoupled parameterization. *The Visual Computer* 2008; **24** (7-9):587-594.
[8] Jang WS, Lee WK, Lee IK, Lee J. Enriching a motion database by analogous combination of partial human motions. *The Visual Computer* 2008; **24** (4):271-280.
[9] Majkowska A, Zordan V, Faloutsos P. Automatic splicing for hand and body animations. Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2006, pp. 309-316.
[10] Ma W, Xia S, Hodgins JK, Yang X, Li C, Wang Z. Modeling style and variation in human motion. Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2010, pp. 21-30.
[11] Tamada K, Kitaoka S, Kitamura Y. Splicing motion graphs: Interactive generation of character animation. Short papers of Computer Graphics International, 2010.
[12] Ng WW, Choy CS, Lun DP, Chau LP. Synchronized partial-body motion graphs. Proc. of ACM SIGGRAPH ASIA, Sketches, 2010, pp. 28.
[13] Horn PKB. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America* 1987;**4**:629-642.
[14] Wang J, Xia S. Layered interpolation for interactive avatar control. Proc. of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry, 2011, pp. 49–58.
[15] Huang Y, Kallmann M. Motion parameterization with inverse blending. Proc. of the 3rd International Conference on Motion in Games, 2010.
[16] Kovar L, Gleicher M. Automated extraction and parameterization of motions in large data sets. *ACM Transaction on Graphics* 2004;**23**(3):559-568.
[17] Carnegie Mellon University, Motion Capture Database, from http://mocap.cs.cmu.edu/, last access 31/08/2013**.**
[18] Mousas C, Newbury P. Real-time motion synthesis for multiple goal-directed tasks using motion layers. Proc. of 9th Workshop on Virtual Reality Interaction and Physical Simulation, 2012, pp. 79-85